

Model Creation

Step 1 : Import Libraries

```
In [7]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
```

Step 2 : Load the dataset and display the first 5 rows

```
In [8]: data = pd.read_csv("C:/Users/lenovo/Desktop/shivamogga_house_cleaned.csv")
data .head()
```

Out[8]:

	Total_SqFt	BHK	Bathrooms	Dist_to_School_km	Dist_to_Hospital_km	Dist_to_Railway
0	1.367597	4	4	-0.281449	-0.406812	-1.18
1	1.255005	4	4	2.584619	1.337755	-1.09
2	1.259438	4	4	1.124977	0.869351	0.40
3	0.922548	4	4	1.140182	1.916091	0.34
4	-1.168829	2	3	-0.760395	0.630369	1.64



```
In [9]: data.drop("Price_per_SqFt",axis=1,inplace=True)
```

```
In [10]: data.head()
```

Out[10]:

	Total_SqFt	BHK	Bathrooms	Dist_to_School_km	Dist_to_Hospital_km	Dist_to_Railway
0	1.367597	4	4	-0.281449	-0.406812	-1.18
1	1.255005	4	4	2.584619	1.337755	-1.09
2	1.259438	4	4	1.124977	0.869351	0.40
3	0.922548	4	4	1.140182	1.916091	0.34
4	-1.168829	2	3	-0.760395	0.630369	1.64



```
In [11]: data.info() #check the info
```

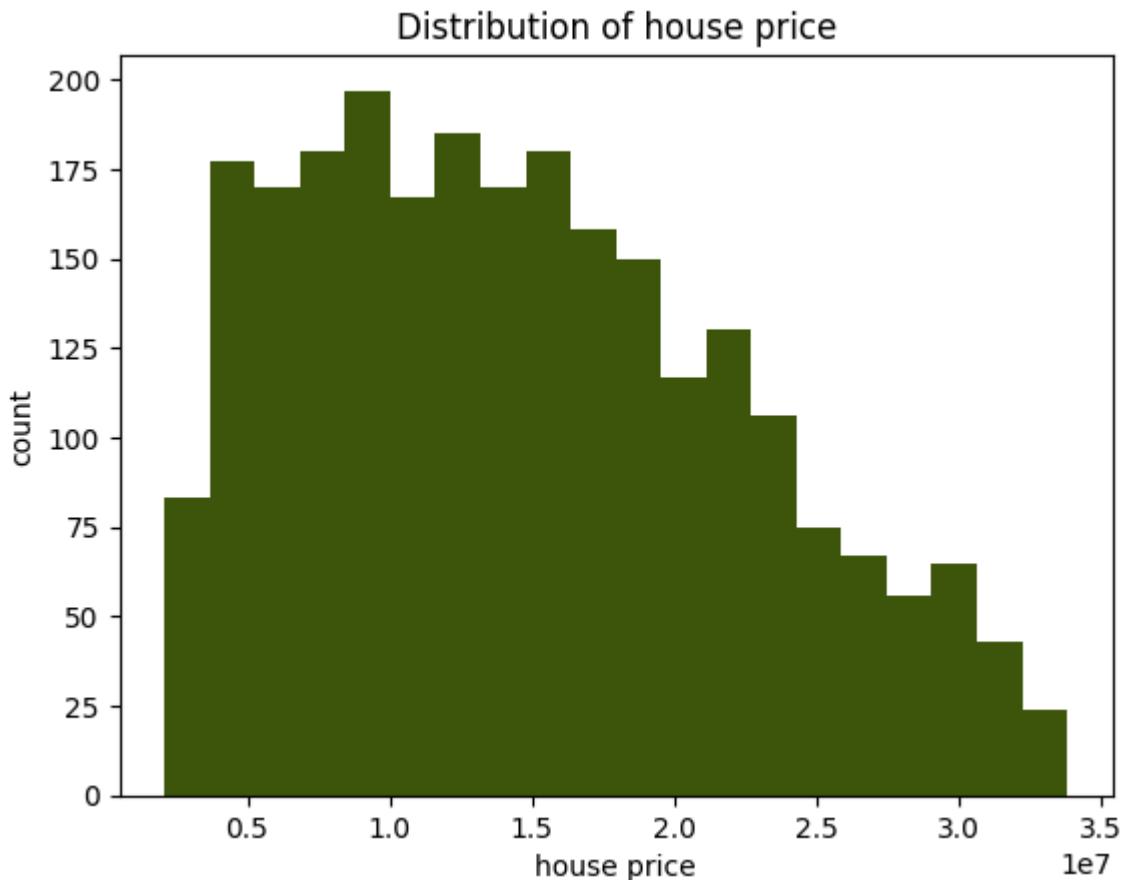
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2500 entries, 0 to 2499
Data columns (total 19 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Total_SqFt        2500 non-null   float64 
 1   BHK               2500 non-null   int64  
 2   Bathrooms         2500 non-null   int64  
 3   Dist_to_School_km 2500 non-null   float64 
 4   Dist_to_Hospital_km 2500 non-null   float64 
 5   Dist_to_Railway_km 2500 non-null   float64 
 6   Parking_Available 2500 non-null   int64  
 7   Gated_Security    2500 non-null   int64  
 8   Price_INR          2500 non-null   int64  
 9   Total_Distance_Score 2500 non-null   float64 
 10  Bath_per_BHK      2500 non-null   float64 
 11  Amenity_Score     2500 non-null   int64  
 12  Locality_Basavanagudi 2500 non-null   int64  
 13  Locality_Bommanakatte 2500 non-null   int64  
 14  Locality_Gadikoppa  2500 non-null   int64  
 15  Locality_Gopala    2500 non-null   int64  
 16  Locality_KHB Colony 2500 non-null   int64  
 17  Locality_Sulebailu  2500 non-null   int64  
 18  Locality_Vinoba Nagar 2500 non-null   int64  
dtypes: float64(6), int64(13)
memory usage: 371.2 KB
```

In [12]: `data.describe() #display the statistical info`

Out[12]:

	Total_SqFt	BHK	Bathrooms	Dist_to_School_km	Dist_to_Hospital_km
count	2.500000e+03	2500.000000	2500.000000	2.500000e+03	2.500000e+03
mean	5.200001e-12	3.108800	3.303600	8.800001e-12	-5.199996e-12
std	1.000200e+00	0.955682	1.045688	1.000200e+00	1.000200e+00
min	-1.740655e+00	1.000000	1.000000	-1.444602e+00	-1.362739e+00
25%	-8.678442e-01	2.000000	3.000000	-6.995763e-01	-7.509458e-01
50%	-3.012504e-03	3.000000	4.000000	-3.042564e-01	-3.159989e-01
75%	8.844263e-01	4.000000	4.000000	4.578747e-01	5.455304e-01
max	1.714239e+00	4.000000	5.000000	2.957132e+00	2.843340e+00

In [13]: `plt.hist(data['Price_INR'], bins=20, color="#3d570e")
plt.xlabel("house price")
plt.ylabel("count")
plt.title("Distribution of house price")
plt.show()`



Step 3 : Split the Feature and Target

```
In [14]: X = data.drop("Price_INR",axis=1)
Y = data["Price_INR"]
```

Step 4 : Split the data into train and test

```
In [15]: X_train , X_test , Y_train , Y_test = train_test_split(X , Y , test_size = 0.2 ,
```

Step 5 : Train the model

```
In [16]: model = RandomForestRegressor(n_estimators=100 , random_state=42)

# fit the model
model.fit(X_train , Y_train)
```

```
Out[16]: ▾ RandomForestRegressor ⓘ ?
```

- ▶ Parameters

Step 6 : predict on test data

```
In [17]: y_pred = model.predict(X_test)
```

Step 7 : Evaluate the model

```
In [18]: mae = mean_absolute_error(Y_test , y_pred)
rmse = np.sqrt(mean_squared_error(Y_test , y_pred))
r2 = r2_score(Y_test , y_pred)
print("mean absolute error :",mae)
print("mean squared error :" ,rmse)
print("r2 score :" ,r2)
```

```
mean absolute error : 270355.08
mean squared error : 388051.38249103044
r2 score : 0.9973777030814557
```

Step 8 : Visualize the Actual vs Predicted value

```
In [19]: plt.scatter(Y_test , y_pred,alpha=0.3 , color="#860f58")
plt.xlabel("Actual Value")
plt.ylabel("Predicted Value")
plt.title("Actual vs Predicted")
plt.show()
```

